# Formal Verification via MCMAS & PRISM

### Hongyang Qu

University of Sheffield

1 December 2015

### Outline

- Motivation for Formal Verification
- Overview of MCMAS
- Overview of PRISM

#### Verification



Developer Verification Agent

### Formal verification

#### It is a systematic way to check all behaviour of a system with respect to certain specification



## Why formal verification is important?

#### Pentium FDIV bug



Logic verification of

critical subsystems

The Explosion of the Ariane 5



\$500 million loss

### Can driverless cars run politely?

### An example in robotics

Two UAVs fly towards each other at the same altitude

- Each UAV has two actions:
  - High altitude and low altitude
- ► UAV 1: action A (high) or B (low)
- ► UAV2: action C (low) and D (high)

	Action C	Action D
Action A	$\checkmark$	×
Action B	×	✓

This scenario can be cast as a game



# MCMAS: A Model Checker for Multi-Agent Systems

- Multi-agent systems are an active research area in Artificial Intelligence (AI).
  - They can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve.
- MCMAS can check complex properties, generate executions leading to bugs and find strategies for game models.

### MCMAS (<u>http://vas.doc.ic.ac.uk/software/mcmas/</u>)

- Symbolic model checker via OBDDs
- Input language ISPL (Interpreted Systems Programming Language)
- Support CTL + Epistemic logic + ATL
- Support (unconditional) fairness
- Efficient implementation of model checking algorithms
- Counterexample/witness generation
- Eclipse-based GUI
- Many applications
- It is actively maintained and developed.

### Ordered Binary Decision Diagram (OBDD)



Truth table of  $\neg a \lor (a \land \neg b \land \neg c)$ 

OBDD of  $\neg a \lor (a \land \neg b \land \neg c)$ 

### Interpreted systems

- An interpreted system IS is composed of N agents  $A = \{1, ..., n\}$
- Each agent  $i \in A$  has
  - a finite set of *local states*  $L_i = \{l_i^1, \dots, l_i^{nl_i}\}$  and
  - a finite set of *actions*  $Act_i = \{a_i^1, \dots, a_i^{na_i}\}$
  - a local protocol  $P_i: L_i \to 2^{Act_i}$
  - an evolution function  $Ev_i: L_i \times Act_1 \times \cdots \times Act_n \to L_i$
- A global state is  $s = (l_1, ..., l_n)$ , and the set of states is S
- A global joint action is  $a = (a_1, ..., a_n)$

## Computation Tree Logic (CTL)

- $\varphi ::= p |\neg \varphi| \varphi \land \varphi$  $|EX\varphi| EG\varphi |EF\varphi| E(\varphi U\varphi)$  $|AX\varphi| AG\varphi |AF\varphi| A(\varphi U\varphi)$ 
  - Path quantifier: *E* (exists) and *A* (all)
  - Temporal operator: X (next), G (globally), U (until) and F (future)



## Computation Tree Logic (CTL)

- $\varphi ::= p |\neg \varphi| \varphi \land \varphi$  $|EX\varphi| EG\varphi |EF\varphi| E(\varphi U\varphi)$  $|AX\varphi| AG\varphi |AF\varphi| A(\varphi U\varphi)$ 
  - Path quantifier: *E* (exists) and *A* (all)
  - Temporal operator: X (next), G (globally), U (until) and F (future)

AGp

AFp

### MCMAS screenshots (1)

Java - mcmas-1.1/examples/Tianji.ispl - Eclipse SDK		
<u>File E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>M</u> CMAS <u>R</u> un <u>W</u> indow <u>H</u> elp		
💼 - 🔄 🖻 💼 🛛 🗊 - 🔅 - 🔉 - 🚱 - 🚱 - 😫 🖶 🞯 - 🤔 💋 🛷 -	i 🖢 👻	· 🖓 • 🍫 🔶 • 🛛 • 📑 🎽
Add agent Content Assist Format source Check syntax Explicit interactive mode Symbolic interacti	ive mode	e Launch verification
⊖ Agent Environment	<u> </u>	Agent Environment
He ODSVars:		▲ Obsvars
b: 03;		a : bounded integer
end Obsvars		b : bounded integer
<pre>     Actions = {null}; </pre>		Actions
Protocol:		Protocol
Other: {null};	=	item 0
Evolution:		▲ Evolution
a=a+1 if (Tianji.Action=Fa and King.Action=Me) or		item 0
(Tianji.Action=Me and King.Action=S1) or		item 1 🗧
(Tianji.Action=Fa and King.Action=S1);		⊿ Agent Tianji
b=b+1 if (Tianji.Action=Fa and King.Action=Fa) or		⊿ Vars
(Tianji Action=Sl and King Action=Sl) or		state : enumeration
(Tianji.Action=Me and King.Action=Fa) or		Actions
(Tianji.Action=Sl and King.Action=Me) or		⊿ Protocol
(Tianji.Action=Sl and King.Action=Fa);		item 0
end Evolution		item 1
end Agent		item 2
⊖ Agent Tianii		item 3
⊖ Vars:		item 4
<pre>state: {FaMeS1, FaMe, MeS1, FaS1, Fa, Me, S1, none};</pre>		item 5
end Vars		item 6
<pre>Actions = {Fa, Me, SI, null}; </pre>		item 7
state=FaMeS1: {Fa. Me. S1}:		▲ Evolution
state=FaMe: {Fa, Me};		item 0
<pre>state=MeS1: {Me, S1};</pre>		item 1
<pre>state=FaS1: {Fa, S1};</pre>		item 2
state=Fa: {Fa};		item 3
state=S1: {S1}:		item 4
stato-poppi (pull)	-	item 5
	•	item 6
Tianji.ispl Interactive Mode   Model Checking   Formula 1		item 7 🔹 🔻
		e 🛷 📮 🎱 🍰

### MCMAS screenshots (2)



### MCMAS screenshots (3)



### Case study: Inconsistent reasoning

- A robot has one sensing event and two decision predicates
  - *a*: sensing event
  - *b*, *c*: predicates
- Reasoning rules:
  - $a \rightarrow \neg b$
  - $a \rightarrow c$
  - $\neg b \rightarrow \neg c$
- Initially, *a* is true, *b* and *c* are unknown

### MCMAS model (1)

Semantics = SingleAssignment; Agent M Vars: a: boolean; b: {unknown, TRUE, FALSE}; c: {unknown, TRUE, FALSE}; end Vars Actions = {none}; Protocol: Other: {none}; end Protocol Evolution: b=FALSE if a=true; c=TRUE if a=true; b=FALSE if c=FALSE; end Evolution end Agent

#### Evaluation

a\_true if M.a=true; a\_false if M.a=false;

b\_true if M.b=TRUE; b\_false if M.b=FALSE; b\_unknown if M.b=unknown;

c\_true if M.c=TRUE; c\_false if M.c=FALSE; c\_unknown if M.c=unknown; end Evaluation

#### InitStates

M.a=true and M.b=unknown and M.c=unknown; end InitStates

### MCMAS model (2)

#### Formulae

AF (((AG a\_true) or (AG a\_false)) and ((AG b\_true) or (AG b\_false) or (AG b\_unknown)) and ((AG c\_true) or (AG c\_false) or (AG c\_unknown)));

AG ((!((EX a\_true) and (EX a\_false))) and (!((EX b\_true) and (EX b\_false))) and (!((EX c\_true) and (EX c\_false)))); end Formulae Formula 1: Eventually all variables won't change their value (become stable)

Formula 2: It is always that no variable can be assigned to different values.

## PRISM (http://www.prismmodelchecker.org/)

- The most popular probabilistic model checker for verifying/analysing systems that have probabilistic behaviour
  - Support rich probabilistic models and specification languages
  - Various verification engines (MTBDD, sparse, hybrid, explicit)
  - State-of-the-art performance
  - Intuitive GUI
  - Actively maintained and developed
  - Has been applied to analyse swarm robots, robot coordination, autonomous systems, and many others.

## Discrete-Time Markov Chains (DTMCs)

- A DTMC is a state-transition system with transitions labelled probabilities
  - A state is a possible configuration of the system
  - Transitions between states represent evolution of the system
  - From a state, the system can move to other states with certain probabilities
- Can be represented as a tuple  $M = (S, \text{Steps}, \overline{s})$  where
  - *S* is a finite set of states
  - $\bar{s} \in S$  is the initial state
  - Steps:  $S \rightarrow Dist(S)$  is a probabilistic transition function
- A DTMC is memoryless, which means the probability distribution in a state does not depend on the history of evolution

### DTMC model for coordination between UAVs



## Other porpular probabilistic models

- Markov Decision Processes (MDP)
  - $M = (S, \Sigma, \text{Steps}, \overline{s})$  where
    - $\Sigma$  is a finite set of actions
    - Steps:  $(S \times \Sigma) \rightarrow Dist(S)$  is a probabilistic transition function
- Continuous-Time Markov Chains (CTMC)
  - $M = (S, R, \overline{s})$  where
    - $R: S \times S \rightarrow \mathbb{R}_{>0}$  is a transition rate matrix

## Probabilistic Specifications

- Reachability properties
  - The probability of reaching a set of states from the initial state
  - Example: A message is delivered successfully with probability 90%.
- Steady state properties
  - The probability of staying in a state (Nash equilibrium) in the long run
  - Example: What is the probability of the queue being 50% full in the long run?
- Reward properties
  - Properties about instantaneous/cumulative rewards attached to states and/or transitions
  - Example: What is the average elapse time of delivering a message?
- Verification of probabilistic properties involves heavy matrix operations (usually multiplications)

### PRISM screenshots (1)

PRISM 3.1								
<u>File E</u> dit <u>M</u> odel <u>P</u> roperties <u>O</u> ptions								
PRISM Model File: /home/luser/tutorial/examples/power_policy1.sm								
PRISM Model File: /home/luser/tutorial/examples/power/p         Model: power_policy1.sm         • Type: Stochastic (CTMC)         • Modules         • & SQ         • a q         • o min: 0         • max: q_max         • init: 0         • max: 2         • init: 0         • max: 2         • init: 0         • max: 2         • init: 0         • a g_max : int         • rate_serve : double         • rate_serve : double         • rate_izs : double         • q_trigger : int	<pre>power_policy1.sm // // Service Queue (SD) // Stores requests which arrive into the system to be processed. // Naximum queue size const int q_max = 20; // Request arrival rate const double rate_arrive = 1/0.72; // (mean inter-arrival time is 0.72 seconds) module SQ // q = number of requests currently in queue q : [0q_max] init 0; // A request arrives [request] true -&gt; rate_arrive : (q'=min(q+1,q_max)); // A request is served [serve] q&gt;1 -&gt; (q'=q-1); // Last request is served [serve_last] q=1 -&gt; (q'=q-1); // Frocesses requests from service queue. // The SP has 3 power states: sleep, idle and busy // Rate of service (average service time = 0.0085) const double rate_serve = 1/0.008; // Rate of switching from sleep to idle (average transition time = 1.65) const double rate_size = 1/0.67; // Rate of switching from idle to sleep (average transition time = 0.675) const double rate (sign = 1.55) const doubl</pre>							
NO OF TRANSITIONS:		-						
Model Properties Simulator Log								
Loading model done.								

## PRISM screenshots (2)



### PRISM screenshots (3)

PRISM 3.1									<u> </u>
<u>File Edit Model Properties Options</u>									
Exploration	Simulation Path								
🚔 Auto Update	1000	Mod	el Type:		Path Length:		Total 1	Cime:	
No Stone: 1	E New Path	Sto	ochastic (CTM	<b>_</b> )	19		145.9	99086646309	85
No. steps.	E* NC00 Fa(II	State	e Rewards:		Transition R	te wards :	Total F	Reward:	
A Do Update	🖁 🛛 🕅 🔀 Reset Path	14	468.5823073	D94,	0.0,		1446	8.582307309	4,
			27063427051 1	JZU96,	0.0, 4 N		1.270	06342705102	196,
State time: 1.0 🖉 Auto	🔒 🟦 Export Path	Defi	ned Constant						
Action Rate Undate		N=	=5,T=100.0						
♦ Left 0.0020 left_n'=0									
Right 0.0080 right_n'=3	Step	left_n	left	right_n	right	r	line	line_n	toleft
ToRight 2.5E-4 toright_n'=fals	0	5	(false)	<u></u>	(false)	(false)	(false)	(true)	(false)
[startLeft] 10.0 left'=true, r'=	a <u>1</u>			4					
[startRight] 10.0 right'=true, r'	2				(true)	(true)			
[startToLeft] 10.0 r'=true, toleft'	3			S	(false)	(false)			
[startLine]  10.0  r'=true, line'=	4							(false)	
	5								
-Path Modification	6								
	2 <u>7</u>	<u> </u>							
	8								
	9								
To Step: 0 Before: 0	10								
	<u>11</u>			4					
	12			<u> </u>					
Formulae	13			Q					
	14		(true)			(true)			
Path formulae:	15		(false)			(false)			
X true U<=T !"minimum"	16				(true)	(true)			
X true U[T,T] !"minimum"	17			3	(false)	(false)			
✓ true U<=T "premium"	18				(true)	(true)			
State labels:	19	1	(false)	4	(false)	(false)	(false)	(false)	(false)
🗙 deadlock 🔷	1000								
🖌 minimum									
🗙 premium 🔍 👻					00000				
		000000000000000000000000000000000000000		000000000000000000000000000000000000000	8888				
Model Properties Simulator Log									
Loading properties done									
Eodang properties done.									

### Case study: swarm aggregation

- The robots have to cluster in one of the two aggregation areas
- The robots go around at random and stop if they encounter a black spot (aggregation area)
- According to a certain probability, they leave the aggregation area and restart walking randomly



### DTMC model

• 
$$p_{ca} = p_{cb} = \frac{S_{agg}}{S_{all}}$$

• 
$$p_{aa} = 1 - p_{ac}, p_{bb} = 1 - p_{bc}, p_{cc} = 1 - p_{ca} - p_{cb}$$

• 
$$p_{ac} = p_{bc} = p_{max} \times (1 - \frac{N_s}{N})$$



## PRISM program (1)

dtmc

const int N = 3;

const double Pca = 0.08;

const double Pcb = Pca;

const double P\_max = 0.2;

formula Pac = P\_max \* (1 - a/N);

formula Pbc = P\_max \* (1 - b/N);

## PRISM program (2)

endmodule

### Probabilistic properties

- Let " areaA " = a = N and "areaB" = b = N;
  - P=? [ F "areaA" | "areaB"]

What is the probability of all robots entering area A or area B?

• S=? [ "areaA"]

In the long run, what is the probability of all robots staying in area A?

### References

- Alessio Lomuscio, Hongyang Qu, Franco Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer (STTT), 2015
- Marta Kwiatkowska, Gethin Norman and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11),* volume 6806 of LNCS, pages 585-591, 2011.